GLOBAL TERRORISM DATABASE (GTD) ANALYSIS

Name: Rudy Martinez

Organization: University of Texas at San Antonio

Course: DA-6813-001-Fall-2021-Data Analytics Applications

Libraries

```
#Data Manipulation
import pandas as pd
import numpy as np

#Data Visualization
import matplotlib.pyplot as plt
import seaborn as sb
#!pip install heatmapz
from heatmap import heatmap, corrplot

#Machine Learning Models
from sklearn.neighbors import KNeighborsClassifier
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import SGDClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline

#Data Processing
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

#Results Evaluation
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, mean_squared_error, r2_score
```

Read Data

```
#Read In Data
data = pd.read_csv("/content/drive/MyDrive/MSDA/Colab Notebooks/Data Applications (Fall 2021)/Individual Project/Data/globalterrorismdb_0221dist.csv")
```

```
#Create Dataframe Copy
terror_df = data.copy()
```

```
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2718: DtypeWarning: Columns (4,31,33,54,61,62,63,76,79,90,92,94,96,114,115,121) hav
  interactivity=interactivity, compiler=compiler, result=result)
```

Data Exploration

```
#Variables in Dataset
print("Number of Variables in Dataset:", len(terror_df.columns))
```

```
Number of Variables in Dataset: 135
```

```
#Records in Dataset
print("Number of Records in Dataset:", len(terror_df))
```

```
Number of Records in Dataset: 201183
```

```
#Variables (Columns)
terror_df.columns.values
```

```
array(['eventid', 'iyear', 'imonth', 'iday', 'approxdate', 'extended',
       'resolution', 'country', 'country_txt', 'region', 'region_txt',
       'provstate', 'city', 'latitude', 'longitude', 'specificity',
       'vicinity', 'location', 'summary', 'crit1', 'crit2', 'crit3',
       'doubtterr', 'alternative', 'alternative_txt', 'multiple',
       'success', 'suicide', 'attacktype1', 'attacktype1_txt',
       'attacktype2', 'attacktype2_txt', 'attacktype3', 'attacktype3_txt',
       'targtype1', 'targtype1_txt', 'targsubtype1', 'targsubtype1_txt',
       'corp1', 'target1', 'natlty1', 'natlty1_txt', 'targtype2',
       'targtype2_txt', 'targsubtype2', 'targsubtype2_txt', 'corp2',
       'target2', 'natlty2', 'natlty2_txt', 'targtype3', 'targtype3_txt',
       'targsubtype3', 'targsubtype3_txt', 'corp3', 'target3', 'natlty3',
       'natlty3_txt', 'gname', 'gsubname', 'gname2', 'gsubname2',
       'gname3', 'gsubname3', 'motive', 'guncertain1', 'guncertain2',
       'guncertain3', 'individual', 'nperps', 'nperpcap', 'claimed',
       'claimmode', 'claimmode_txt', 'claim2', 'claimmode2',
       'claimmode2_txt', 'claim3', 'claimmode3', 'claimmode3_txt',
       'compclaim', 'weaptype1', 'weaptype1_txt', 'weapsubtype1',
       'weapsubtype1_txt', 'weaptype2', 'weaptype2_txt', 'weapsubtype2',
       'weapsubtype2_txt', 'weaptype3', 'weaptype3_txt', 'weapsubtype3',
       'weapsubtype3_txt', 'weaptype4', 'weaptype4_txt', 'weapsubtype4',
       'weapsubtype4_txt', 'weapdetail', 'nkill', 'nkillus', 'nkillter',
       'nwound', 'nwoundus', 'nwoundte', 'property', 'propextent',
       'propextent_txt', 'propvalue', 'propcomment', 'ishostkid',
       'nhostkid', 'nhostkidus', 'nhours', 'ndays', 'divert',
       'kidhijcountry', 'ransom', 'ransomamt', 'ransomamtus',
       'ransompaid', 'ransompaidus', 'ransomnote', 'hostkidoutcome',
       'hostkidoutcome_txt', 'nreleased', 'addnotes', 'scite1', 'scite2',
```

```
            'scite3', 'dbsource', 'INT_LOG', 'INT_IDEO', 'INT_MISC', 'INT_ANY',
            'related'], dtype=object)
```

```
#Add Additional Data Exploration of:
## - Number of Terrorist Attacks per Year
## - Number of Terrorist Attacks by Region
## - Number of Terrorist Attacks by Region by Year
## - Number of Terrorist Groups that Have Committed Various Ranges of Attacks (Increments of 20) (How many groups have attached within different ranges of attac
## - Number of Attacks per Unique Terrorist Group (Only Terrorist Groups with More Than the Average of Terrorist Attacks)
## - Number of Attacks for Top 5 Groups Over the Length of Time in the Dataset
## - Most Common Attack Types
## - Most Commons Attack Types and Number of Deaths per Attack Type
```

Data Cleaning

```
#Drop Duplicate Rows from Dataset
terror_df = terror_df.drop_duplicates()

#Records in Dataset
print("Number of Records in Dataset:", len(terror_df))
```

```
Number of Records in Dataset: 201183
```

- **Result:** No Duplicate Values - Proceed with Additional Data Cleaning

```
#Null Values in Dataset
null_values = terror_df.isnull().sum()
null_values = pd.DataFrame(null_values, index = None, columns=["Null Count"])

#Null Count Greater Than Zero
greater_than_zero = null_values[null_values["Null Count"] > 0]
null_variables = len(greater_than_zero)
print(null_variables, "Variables with Null Values in the Dataset")
```

```
103 Variables with Null Values in the Dataset
```

```
#Create Dataframe with Columns and Their Nulls
null_values.reset_index(inplace=True)
null_values = null_values.rename(columns = {'index':'Variable'})
null_values.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 135 entries, 0 to 134
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
```

```
 0   Variable    135 non-null    object
 1   Null Count  135 non-null    int64 
dtypes: int64(1), object(1)
memory usage: 2.2+ KB
```

```
#Calculate Percentage Missing and Percentage Complete
null_values["Percentage_Missing"] = null_values["Null Count"] / (len(terror_df))
null_values["Percentage_Complete"] = 1 - null_values["Percentage_Missing"]


#Select Attributes with a completeness above 95%
perc_complete = .95
selected_attributes = null_values[null_values["Percentage_Complete"] >= perc_complete]
print(len(selected_attributes), f"Attributes Have a Percentage_Complete >= {int(perc_complete*100)}%")
```

```
41 Attributes Have a Percentage_Complete >= 95%
```

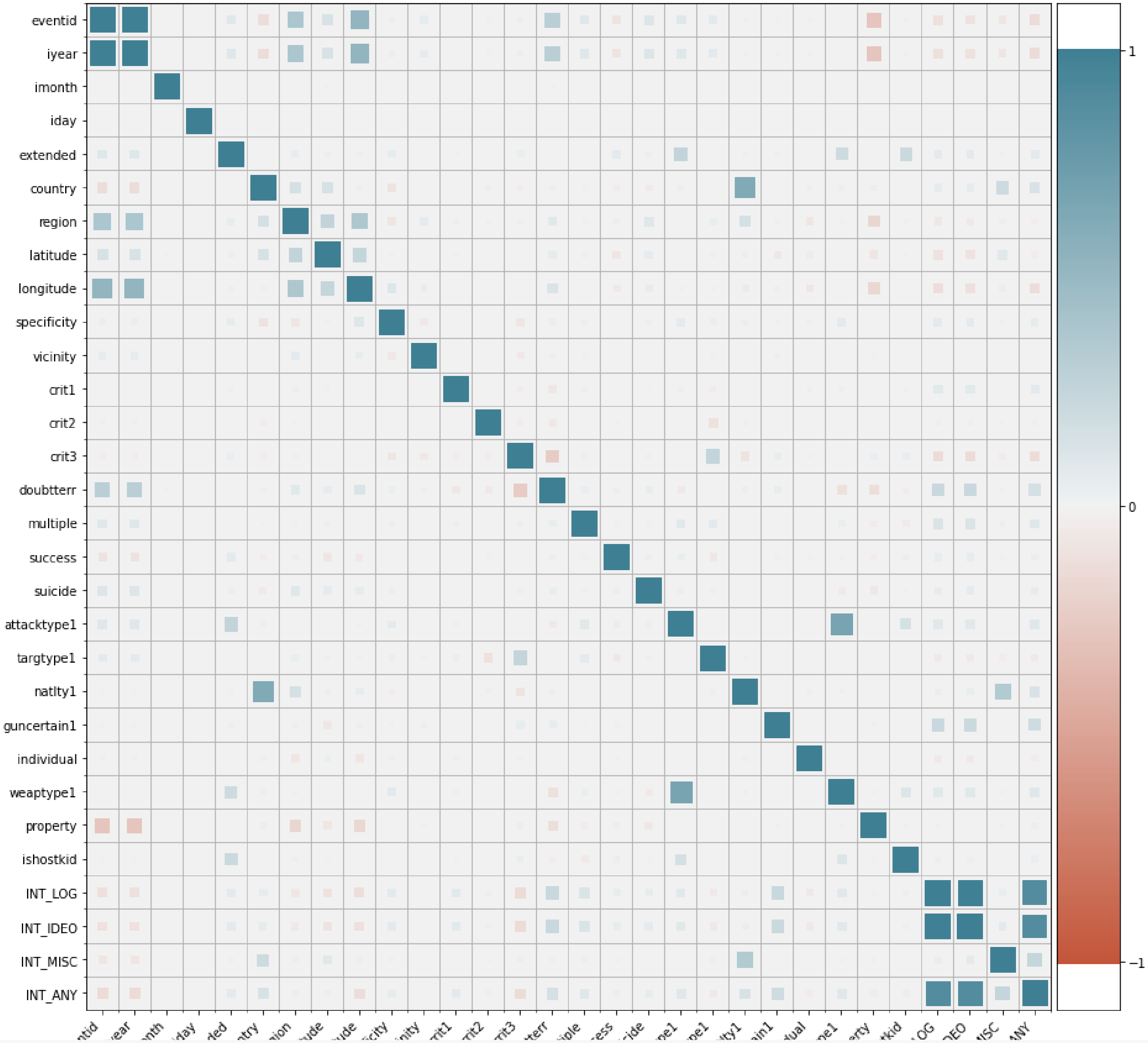
```
## Cut terror_df to include only selected attributes
selected_attributes_list = selected_attributes["Variable"].to_list()
terror_df_cleaned = terror_df.loc[:, terror_df.columns.isin(selected_attributes_list)]
terror_df_cleaned.columns
```

```
Index(['eventid', 'iyear', 'imonth', 'iday', 'extended', 'country',
       'country_txt', 'region', 'region_txt', 'provstate', 'city', 'latitude',
       'longitude', 'specificity', 'vicinity', 'crit1', 'crit2', 'crit3',
       'doubtterr', 'multiple', 'success', 'suicide', 'attacktype1',
       'attacktype1_txt', 'targtype1', 'targtype1_txt', 'target1', 'natlty1',
       'natlty1_txt', 'gname', 'guncertain1', 'individual', 'weaptype1',
       'weaptype1_txt', 'property', 'ishostkid', 'dbsource', 'INT_LOG',
       'INT_IDEO', 'INT_MISC', 'INT_ANY'],
      dtype='object')
```

- **Data Cleaning Process and Results:**
    - Following the Data Cleaning, the original dataset that contained 135 attributes was narrowed down to 41 attributes. This was accomplished by identifying attributes that have a percentage complete between 95% and 100%. This was calculated by measuring the null count across all variables.

Data Preprocessing

```
#Variable Correlations of Remaining Variables
plt.figure(figsize=(15, 15))
corrplot(terror_df_cleaned.corr(), size_scale=400)
```




```
correlations = terror_df_cleaned.corr()
correlations
```

	eventid	iyear	imonth	iday	extended	country	region	latitude	longitude	specificity	vicinity	crit1	crit2	crit3	doubtt
eventid	1.000000	0.999996	-0.002286	0.018233	0.099490	-0.127593	0.393426	0.143636	0.536555	0.042471	0.063482	0.000320	0.024338	-0.032390	0.310
iyear	0.999996	1.000000	-0.004820	0.018148	0.099480	-0.127568	0.393442	0.143672	0.536558	0.042452	0.063499	0.000303	0.024333	-0.032387	0.310
imonth	-0.002286	-0.004820	1.000000	0.006866	0.000359	-0.009091	-0.006155	-0.012712	-0.003412	0.005554	-0.006560	-0.000037	0.001854	0.000312	-0.012
iday	0.018233	0.018148	0.006866	1.000000	-0.004845	0.002492	0.008363	0.002965	0.012975	-0.005938	-0.005250	0.009995	-0.001912	-0.004270	0.002
extended	0.099490	0.099480	0.000359	-0.004845	1.000000	-0.010907	0.052668	-0.033986	0.029282	0.059299	0.015101	-0.011856	0.001785	0.048839	0.014
country	-0.127593	-0.127568	-0.009091	0.002492	-0.010907	1.000000	0.160535	0.150347	-0.020014	-0.084994	-0.009746	-0.010616	-0.037546	-0.036943	0.035
region	0.393426	0.393442	-0.006155	0.008363	0.052668	0.160535	1.000000	0.267367	0.373260	-0.074957	0.070270	0.029147	-0.015348	-0.020332	0.092
latitude	0.143636	0.143672	-0.012712	0.002965	-0.033986	0.150347	0.267367	1.000000	0.253286	-0.016745	0.004116	0.023247	-0.007015	0.001093	0.063
longitude	0.536555	0.536558	-0.003412	0.012975	0.029282	-0.020014	0.373260	0.253286	1.000000	0.110228	0.052155	-0.003131	-0.004225	0.007526	0.127
specificity	0.042471	0.042452	0.005554	-0.005938	0.059299	-0.084994	-0.074957	-0.016745	0.110228	1.000000	-0.058349	0.015940	0.003691	-0.073662	0.044
vicinity	0.063482	0.063499	-0.006560	-0.005250	0.015101	-0.009746	0.070270	0.004116	0.052155	-0.058349	1.000000	0.003701	0.000251	-0.055271	0.034
crit1	0.000320	0.000303	-0.000037	0.009995	-0.011856	-0.010616	0.029147	0.023247	-0.003131	0.015940	0.003701	1.000000	-0.009039	-0.041864	-0.059
crit2	0.024338	0.024333	0.001854	-0.001912	0.001785	-0.037546	-0.015348	-0.007015	-0.004225	0.003691	0.000251	-0.009039	1.000000	-0.032268	-0.051
crit3	-0.032390	-0.032387	0.000312	-0.004270	0.048839	-0.036943	-0.020332	0.001093	0.007526	-0.073662	-0.055271	-0.041864	-0.032268	1.000000	-0.238
doubtterr	0.310826	0.310855	-0.012918	0.002694	0.014839	0.035049	0.092268	0.063381	0.127574	0.044406	0.034142	-0.059710	-0.051507	-0.238663	1.000
multiple	0.098638	0.098651	-0.004031	0.000453	-0.007352	-0.016055	0.020150	-0.025230	0.000204	0.038342	-0.011905	0.031653	0.011457	0.039800	0.047
success	-0.087280	-0.087280	0.000971	-0.010885	0.079273	-0.038225	-0.027931	-0.068169	-0.053260	0.009654	-0.002769	-0.009881	-0.013518	-0.002562	-0.031
suicide	0.121448	0.121449	-0.000513	0.003734	-0.035816	-0.049535	0.104998	0.067680	0.059669	-0.034569	0.007940	0.018976	-0.003119	-0.023594	0.053
attacktype1	0.099387	0.099358	0.010125	-0.002347	0.266168	-0.028391	0.016649	0.012022	0.019723	0.070733	0.004813	0.030476	0.006524	0.008210	-0.051
targtype1	0.076574	0.076578	-0.001877	-0.000875	0.015987	-0.008415	0.048313	-0.022046	0.023720	0.041998	0.021971	-0.035624	-0.101254	0.261268	0.001
natlty1	-0.016563	-0.016534	-0.010306	0.002996	0.022529	0.623813	0.168014	-0.030234	0.056817	-0.036193	0.016507	-0.011432	-0.021110	-0.082456	0.038
guncertain1	0.020411	0.020415	-0.002864	-0.001461	0.027682	-0.007836	-0.025632	-0.062618	0.032228	-0.012205	0.029446	0.006242	-0.002867	0.058265	0.054
individual	0.022744	0.022749	-0.001828	-0.002073	-0.011038	0.031551	-0.067665	0.044798	-0.068383	-0.026151	-0.008608	-0.023889	0.002984	0.019914	0.009
weaptype1	0.018518	0.018489	0.011371	-0.000885	0.204190	-0.030338	0.020118	-0.003991	-0.005909	0.078850	-0.000178	0.031826	-0.004731	-0.003827	-0.109
property	-0.284128	-0.284104	-0.009463	-0.006334	0.002645	0.040621	-0.173277	-0.079089	-0.165105	0.012912	-0.014555	-0.011097	-0.004892	0.035637	-0.116
ishostkid	-0.013809	-0.013824	0.004512	0.003682	0.219958	-0.007985	-0.024272	-0.015996	-0.007256	0.014658	0.003439	-0.012570	-0.001555	0.045088	-0.028
INT_LOG	-0.108571	-0.108567	-0.003422	0.000215	0.075577	0.061098	-0.066060	-0.092582	-0.114735	0.078825	0.013137	0.089278	-0.022656	-0.139373	0.224
INT_IDEO	-0.098899	-0.098894	-0.003428	0.000139	0.078431	0.059215	-0.054911	-0.086865	-0.110065	0.076895	0.014127	0.089225	-0.022763	-0.142700	0.226

- **Correlation Analysis:**
    - After identifying the 41 attributes that made it past the data cleaning stage, it is essential to view attribute correlations. This will aid in selecting features for the model.

- The code below quickly identifies the attributes with absolute correlations greater than a threshold of 0.5. These attributes will be removed from our dataset before building the model.

```
corr_pairs = correlations.unstack()
corr_pairs = corr_pairs.abs()
corr_pairs_remove = corr_pairs[corr_pairs > 0.5]
corr_pairs_remove = corr_pairs_remove[corr_pairs_remove < 1.0]
corr_pairs_remove
```

```
eventid      iyear          0.999996
             longitude      0.536555
iyear        eventid        0.999996
             longitude      0.536558
country      natlty1        0.623813
longitude    eventid        0.536555
             iyear          0.536558
attacktype1  weaptype1      0.682546
natlty1      country        0.623813
weaptype1    attacktype1    0.682546
INT_LOG      INT_IDEO       0.996344
             INT_ANY        0.894842
INT_IDEO     INT_LOG        0.996344
             INT_ANY        0.897571
INT_ANY      INT_LOG        0.894842
             INT_IDEO       0.897571
dtype: float64
```

```
#Create list of variables to drop
corr_pairs_remove_df = pd.DataFrame(corr_pairs_remove)
corr_pairs_remove_df.reset_index(inplace=True)
corr_pairs_remove_df = corr_pairs_remove_df.rename(columns = {'level_0':'Var1', 'level_1':'Var2'})

drop_list = set()

for var1, var2 in zip(corr_pairs_remove_df["Var1"], corr_pairs_remove_df["Var2"]):
  drop_list.add(var1)
  drop_list.add(var2)

drop_list
```

```
{'INT_ANY',
 'INT_IDEO',
 'INT_LOG',
 'attacktype1',
 'country',
 'eventid',
 'iyear',
 'longitude',
 'natlty1',
 'weaptype1'}
```

```
#Remove Correlated Variables from Dataframe
terror_df_cleaned = terror_df_cleaned.drop(labels=drop_list, axis=1)
terror_df_cleaned.columns
```

```
Index(['imonth', 'iday', 'extended', 'country_txt', 'region', 'region_txt',
       'provstate', 'city', 'latitude', 'specificity', 'vicinity', 'crit1',
       'crit2', 'crit3', 'doubtterr', 'multiple', 'success', 'suicide',
       'attacktype1_txt', 'targtype1', 'targtype1_txt', 'target1',
       'natlty1_txt', 'gname', 'guncertain1', 'individual', 'weaptype1_txt',
       'property', 'ishostkid', 'dbsource', 'INT_MISC'],
      dtype='object')
```

```
#Check New Count of Attributes
len(terror_df_cleaned.columns)
```

```
31
```

- **Removed Correlated Variables:**
  - There are now 31 variables left in the `terror_df_cleaned` dataframe.

```
terror_df_cleaned = terror_df_cleaned.drop(labels="dbsource", axis=1)
```

- In exploring `dbsource` closer, this variable does not seem relevant to the modeling process as it is simply the database source. This variable will be dropped before proceeding.

- **Target Variable Analysis Steps:**
  - Before building the models, it is important to also evaluate the target variable `gname`. This represents the terrorist organization responsible for the attack.
  - Identify the groups that are responsible for more than 3 attacks
  - Remove rows in the data where the groups below the average are located

```
#View Info on Target
gname_breakdown = terror_df_cleaned.groupby(["gname"])[["gname"]].count().rename(columns={"gname":"Count"})

gname_total = gname_breakdown.index
gname_above_three = gname_breakdown[gname_breakdown["Count"] > 3]

print("Total Groups:", len(gname_total))
print("Count of Groups Responsible for More Than 3 Attacks:", len(gname_above_three))
```

```
Total Groups: 3671
Count of Groups Responsible for More Than 3 Attacks: 1150
```

```
## Cut terror_df_cleaned to include only selected terrorist groups
gname_above_three_list = gname_above_three.index.to_list()
terror_df_cleaned = terror_df_cleaned[terror_df_cleaned["gname"].isin(gname_above_three_list)]
```

```
#View Info on Target After Removal
gname_breakdown = terror_df_cleaned.groupby(["gname"])[["gname"]].count().rename(columns={"gname":"Count"})

gname_total = gname_breakdown.index
```

```
print(" Total Groups in New Dataset:", len(gname_total))
print("\n", "Total Number of Records After Processing", len(terror_df_cleaned))
```

```
 Total Groups in New Dataset: 1150

 Total Number of Records After Processing 197642
```

Additional Preprocessing Steps:

- A final step ahead of modeling is to preprocess categorical variables in the dataset.

```
#Identify Categorical Variables
cat_vars = terror_df_cleaned.loc[:, terror_df_cleaned.dtypes == object]
cat_vars_counts = cat_vars.nunique()
cat_vars_counts
```

```
country_txt          200
region_txt            12
provstate           2548
city               41760
attacktype1_txt        9
targtype1_txt         22
target1            89267
natlty1_txt          213
gname               1150
weaptype1_txt         12
dtype: int64
```

- Based on these results, the number of levels for each categorical variable is clear. Because some of these variables have a large number of levels, only those with levels less than 30 will be included in the model.
- This will ensure Python doesn't crash while trying to create dummy variables.

```
terror_df_cleaned = terror_df_cleaned.drop(labels=["country_txt", "provstate", "city", "target1", "natlty1_txt", ], axis=1)
```

- Now that these variables have been dropped, we must assess null values in the dataset and drop the records in which there are nulls.

```
terror_df_cleaned.isnull().sum()
```

```
imonth              0
iday                0
extended            0
region              0
```

```
region_txt            0
latitude           4509
specificity           1
vicinity              0
crit1                 0
crit2                 0
crit3                 0
doubtterr             0
multiple              0
success               0
suicide               0
attacktype1_txt       0
targtype1             0
targtype1_txt         0
gname                 0
guncertain1         378
individual            0
weaptype1_txt         0
property              0
ishostkid           162
INT_MISC              0
dtype: int64
```

```
terror_df_cleaned = terror_df_cleaned.dropna()
terror_df_cleaned.shape
```

```
(192600, 25)
```

- With this step complete, it is time to OneHotEncode the remaining categorical variables.

```
#OneHotEncoding - Initialize Encoder
encoder = OneHotEncoder(sparse=False)

#Specify Categorical Column
columns = ['region_txt','attacktype1_txt','targtype1_txt','weaptype1_txt']

#Apply Encoder
df_encoded = pd.DataFrame(encoder.fit_transform(terror_df_cleaned[columns]))
df_encoded.columns = encoder.get_feature_names(columns)

#Remove Columns
terror_df_cleaned.drop(columns ,axis=1, inplace=True)

#Reset and Drop Index
terror_df_cleaned.reset_index(drop=True, inplace=True)

#Concatenate the OneHotEncoded Columns
terror_df_cleaned = pd.concat([terror_df_cleaned, df_encoded], axis=1)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function get_feature_names is deprecated; get_feature_names is depre
  warnings.warn(msg, category=FutureWarning)
```

```
#Print Shape and First Five Rows
print("Data Shape: ",terror_df_cleaned.shape)
```

```
Data Shape:  (192600, 76)
```

Modeling

```
#Create Feature Matrix and Labels
feature_matrix = terror_df_cleaned.drop("gname", axis=1)
label = terror_df_cleaned["gname"]

#Split the Data Using train_test_split
X_train, X_test, y_train, y_test = train_test_split(feature_matrix, label, test_size = 0.2, random_state = 123)

#Train and Test Shape
print('Train: X=%s, y=%s' % (X_train.shape, y_train.shape))
print('Test: X=%s, y=%s' % (X_test.shape, y_test.shape))
```

```
Train: X=(154080, 75), y=(154080,)
Test: X=(38520, 75), y=(38520,)
```

Decision Tree Model

```
#Train Model
decision_tree_model = DecisionTreeClassifier(random_state=123)
decision_tree_model.fit(X_train,y_train)

#Generate Predictions
tree_predict = decision_tree_model.predict(X_test)
```

```
#Validation
tree_acc_score = accuracy_score(y_test,tree_predict)
print("Accuracy Score:",tree_acc_score)
```

```
Accuracy Score: 0.6332294911734164
```

K Nearest Neighbors

```
#Train Model
knn = KNeighborsClassifier(n_neighbors=3)
```

```
knn.fit(X_train, y_train)

#Generate Predictions
knn_pred = knn.predict(X_test)
```

```
#Validation
knn_acc_score = accuracy_score(y_test,knn_pred)
print("Accuracy Score:",knn_acc_score)
```

```
Accuracy Score: 0.5796469366562824
```

SGDClassifier

```
#Train Model
sgd_clf = make_pipeline(StandardScaler(), SGDClassifier(max_iter=1000, tol=1e-3))
sgd_clf.fit(X_train, y_train)

sgd_pred = sgd_clf.predict(X_test)
```

```
#Validation
sgd_acc_score = accuracy_score(y_test,sgd_pred)
print("Accuracy Score:",sgd_acc_score)
```

```
Accuracy Score: 0.5745327102803738
```